In Proc. Fourth Int. Conf. on Collaborative Virtual Environments, ACM (Bonn, Germany, Sept. 2002).

Security of Runtime Extensible Virtual Environments

Ernesto J. Sallés, James Bret Michael, Michael Capps, Don McGregor, and Andrzej Kapolka
Naval Postgraduate School, Monterey, California
{ejsalles, bmichael, mcapps, mcgredo, akapolka} @nps.navy.mil

ABSTRACT
Distributed, real-time virtual environment (VE) architectures have traditionally been driven by quality of service (QOS) considerations, with little or no concern paid to security issues. With recent advancements in functionality, computing power and network bandwidth it has become practical to use VEs in sensitive areas such as product development with proprietary information and visualization of classified information. Consequently, previously ignored aspects of security need to be made a primary concern at the outset of designing a VE. In this paper we explore security concerns associated with a subtype of VEs: Runtime Extensible VEs (RTEVEs). We introduce a taxonomy of security issues, derived from a case study of NPSNET-V, with the goal of using this taxonomy to guide the formulation of security policy, requirements, and architectures for RTEVEs.

Categories and Subject Descriptors
K.6.5 [**Management of Computing and Information Systems**]: Security and Protection.

General Terms: Security

Keywords: computer security, information assurance, virtual reality, virtual environment, collaborative environment, runtime extensible virtual environment, taxonomy, security.

1. INTRODUCTION
Researchers affiliated with the MOVES Institute of the Naval Postgraduate School are developing a RTEVE known as NPSNET-V [4]. RTEVEs are a subset of VEs and primarily differ from general VEs in that they can dynamically load new components at runtime. Previous versions of NPSNET lacked this capability, as they were used to research a variety of other topics in networked graphics, such as performance; and, as is often the case with research projects, little attention was given to security.

VEs in general are moving out of the laboratory environment and into commercial use, thus necessitating explicit consideration of security in all phases of their development. The dynamic loading of code-containing components—the defining aspect of RTEVEs—introduces challenging issues related to security. In this paper, we present a taxonomy of security issues associated with RTEVEs, using NPSNET-V as the primary basis from which to identify and assess security issues.

(N.B.: All references to VEs and RTEVEs in this paper are strictly in the "visual" context in which participants are interacting with other participants in real-time; and their representations are depicted visually in a two- or three-dimensional world.)

CVE'02, September 30–October 2, 2002, Bonn, Germany.
ACM 1-58113-489-4/02/0009.

1.1 Proliferation of VEs
Recent advances in computing technology have made it technically feasible to field near real-time, widely distributed, interactive VEs. Furthermore, a greater number of organizations are beginning to realize what gaming companies and the military have known for years: VEs are more than just research tools; they can be used to gain a competitive advantage, generate revenue, or reduce operational and developmental costs. For example, development of systems and the training of personnel can be performed without placing humans and the real systems in harms way (e.g., virtual prototyping and combat flight training). VE applications include multi-user interactive games, training, product prototyping, entertainment, and decision-making.

1.2 Need for Security within VEs
As VEs proliferate, many will be used in contexts in which there are incentives for malicious users to misuse such systems for their own gain. Manufacturers may use VEs for virtual prototyping, allowing engineers at several locations to make design decisions in a collaborative environment in which they are all virtually present and observe proprietary information. In this case, a competitor engaging in industrial espionage might be able to view the proprietary information and use it to the competitor's advantage in the marketplace, or modify or destroy the information to mislead or disrupt their competitors.

In a different case, a military commander may utilize a VE to visualize the battlefield and all pertinent intelligence information during a conflict. An adversary that is able to exploit weaknesses in the VE could inject false information, leading the commander into make misinformed decisions, such as mistakenly sending troops into an orchestrated ambush.

1.3 Differing Levels of Security
VEs vary in the degree of security they require: this is driven primarily by the organization using the VE and the context in which the VE will be used, with consideration given to tradeoffs such as the cost to develop and maintain the system, system performance, and the risks associated with a violation of security policy or breech of trust, assuming such policy and trust relationships have been established. Some VEs will need high levels of security, running on trusted systems with mandatory access control policies, while others will require minimal-to-medium levels of security controls because the risks associated with the misuse of these systems is considered to be low.

An example of a VE application that might require low levels of security might be an unclassified technical trainer for tasks such

as vehicle repair. A VE intended for use by battlefield commanders in formulating strategy and tactics would require higher levels of security. VE applications requiring the highest levels of security would be those used as part of national strategic systems (e.g., VEs used in support of the Ballistic Missile Defense System).

Other dimensions of VE security are multilevel security (MLS) and compartmentalization. For instance, a military application of a VE could be designed to allow senior commanders to possess sensitive intelligence information while denying it to personnel at lower levels of the chain–of–command, even though both are present in the same area of the virtual environment.

2. WHAT IS A VIRTUAL ENVIRONMENT?

According to Singhal and Zyda [21],

> A networked virtual environment is a software system in which multiple users interact with each other in real-time, even though these users may be located around the world.

VEs require the ability for participants to communicate and "physically" interact within the environment; the feeling of a shared space and time are prerequisites.

In a VE, each user controls one or more entities. Each entity is represented within the VE by a visual model that all other users can see and interact with. The VE delineates the level of realism and interaction through its physics-based model foundation and rules of interaction that are designed into the system.

In this section we cover the basics of the design and architecture of distributed VEs. For a detailed discussion of VE architecture and design please see [5], [16], and [21].

2.1 Characteristics of an Effective VE

Capps and Stotts [5] list attributes of an effective VE architecture, classifying them into the following categories: network topology, interoperability, composability, and rapid evolution. In 1997, they stated that it was difficult to simultaneously address attributes in all four categories, and that the then-current examples of VE architectures were deficient to some degree in one or more of the categories. This evaluation still holds true today, and may continue to for many more years to come.

Network Topology. A good topology will allow for large, if not infinite, scalability in the number of participants in the VE. If any network resource is lost, it will allow for a graceful degradation of the simulation. In addition, it will allow for adequate performance for each participant no matter what type of communication capabilities they possess (e.g., T1, ADSL, modem).

Interoperability. A good attribute of a VE would be the ability to transfer an object to another VE without the loss of information. Control of the object must be transferable between different VEs, to include the physical properties and behaviors of the object. If an object explodes in one VE after a certain sequence of events, then the same object should be able to explode in the other VE.

Composability. It should be possible to take two VEs with desirable attributes, and compose them into a third VE that maintains the unique functionalities of the two parent VEs. The resulting

VE would literally be a union of the two original VEs in every way, both at start-up and during runtime, without undesirable emergent properties.

Rapid Evolution. It should be possible to rapidly incorporate new technology into a VE with a minimal degree of modification to existing components; for example, creating a module with the new desired behavior and simply adding the module to the VE.

To date only the attributes associated with network topology have been reasonably well implemented.

2.2 VE Architectures

In general, a distributed VE requires copies of a VE application residing on a number of workstations connected by a network that is used for passing data. The data that is to be shared may be a combination of administrative communications, entity-data updates, and streaming video and audio. The architecture must allow for data sharing and operation within the QOS constraints, as described in Section 2.3. We now turn to a minimal architecture for a VE, consisting of four components: workstation, application, database, and network.

Workstation. Every VE application must run on a workstation, each one equipped for networked multimedia capabilities (e.g., network connection, graphics card, and sound card).

Application. The application must be able to accurately maintain state information for however many entities are present within the area of view of the host entity of that application. It must correctly maintain each entity's state, perform the actions triggered by input data, and display views of the environment. It must also manage the transmission and reception of data and support communications.

Database. The application must have access to a world description database. This information is used to create the environment (e.g., terrain, structures) and every possible object that can exist. Ownership and responsibility for this data can be centralized or distributed across multiple workstations.

Network. The design of the network topology is crucial to the workings of the VE. Numerous architectures and protocols have been developed, each with strengths and weaknesses. A good topology is application dependent. In general, there will be administrative processes that require a server-based structure with reliable TCP/IP communications. Likewise, entity-state updates that require a constant transmission of data packets between all participants might find a multicast protocol useful, despite the inherent unreliability of common multicast implementations, since occasional dropped packets would likely be quickly outdated by more new data. However, there may be VEs that require reliability of all data transmissions, calling for the use of TCP rather than multicast.

2.3 QOS Concerns for VEs

Here we give an overview of QOS concerns for VEs, with the aim of setting the stage for discussing the relationship between security and other QOS parameters. A goal for architecting and designing VEs is to create an environment in which multiple participants can interact on a near real-time basis; this involves creating the illusion of real-time interaction among entities or objects. The generally accepted standard for frame rate (i.e., the rate at which entities are updated) within a VE is 30 Hz. At lower rates the human eye begins to notice non-continuous motion; thus entity updates are determined at this minimum rate. However, the update rate can be relaxed when dead-reckoning algorithms are used. We note that an essential part of a distributed VE is the data-up-

date packet, which conveys the state information for each entity. The transfer rate of these packets is directly affected by three networking QOS concerns: bandwidth, reliability, and latency and delay.

Bandwidth. The number of bytes per second of information that can be transmitted. Generally speaking, more entities or higher fidelity in a VE requires more bandwidth. Some VEs include interactive audio or streaming video, which add to the bandwidth consumption. Denial-of-service (DOS) attacks have been used extensively in many types of distributed systems to reduce the amount of available bandwidth and any point in time.

Reliability. Ensuring that data is forwarded to the correct destination with an acceptable packet-loss rate. Network and computer survivability have the greatest impact in this area. Reliability of the network is crucial to the sustained performance of the VE, as is the packet-loss rate in congested networks.

Latency/Delay. Minimization of latency and delay is necessary to maintain an acceptable level of synchronization among the participants of a VE. It is difficult to create the illusion of entities being in the same state on all machines because of network-based latencies: an acceptable synchronization window must be established that will provide the desired level of world-consistency.

2.4 Special Case: RTEVEs

Traditional VEs are static; that is, once the VE is running it cannot incorporate new types of entities. To load a new type of entity the application must be stopped, the entity added to the environment, and the application restarted. In contrast, a RTEVE permits the runtime introduction of new objects and behaviors; this trait is essential for a VE that cannot be halted to update the base of objects and behaviors. This ability goes beyond just discovering objects. Potentially, any piece of code can be loaded into the VE application, including new capabilities for the VE itself. To date, the only visual RTEVEs in existence are hosted within research institutions.

The idea here is that as an application has need for a capability or object that it does not possess, it will access a database that contains the necessary data and code modules. This includes modules of executable code that are used to describe model behaviors, communication methods, and system capabilities. This is where the true danger of a RTEVE lies. Any module that may have been maliciously modified on one system could easily be propagated or distributed to thousands of users.

3. SECURITY

Our discussion of security issues associated with RTEVEs begins with a high-level treatment of computer security by discussing prior work, then presenting scenarios of possible events that could occur within RTEVEs and VEs. Prior work, like that discussed in Section 3.1, identifies issues that transcend systems or address specific areas of systems; we prefer to take a system view of RTEVE security.

The five main areas of information assurance—secrecy, integrity, availability, non-repudiation, and authentication—are all concerns throughout the life cycle of a VE; see Table 1. For instance, it may be necessary for a VE, used by battle commanders for tactical-level decision-making, to ensure that participants cannot repudiate the fact that they injected certain data into the VE. Similarly, it is necessary to evaluate system design and maintenance decisions in terms of their potential affect on preventing malicious users of a VE from violating security policies regarding confidentiality or data integrity, masquerading as a legitimate user, or launching a distributed DOS attack.

Table 1. Areas of Information Assurance

Areas	Description
Integrity	Prevent unauthorized modification of data
Confidentiality	Prevent unauthorized viewing of data
Availability	Ensure data is available for its intended use
Non-repudiation	Ensure that a user cannot refute information they placed into the system
Authentication	Ensuring a user/module is who it says it is

In the setting of security policy and implementation of security mechanisms for a VE, consideration should be given to their impact on the QOS concerns previously mentioned. For example, any form of encryption will have some latency effect on the transmission of data packets. Using public-key cryptography for state updates would be much more computationally intensive than using conventional, single-key cryptography. However, even symmetric-key cryptography will introduce some latency.

3.1 Security Efforts to Date

Much research has been conducted on computer security and information assurance. However, the complexity of information systems such as VEs and the resourcefulness and perseverance of rogue users of such systems are constant drivers for conducting additional research (e.g., making it more difficult to circumvent digital watermarking algorithms). General network and computer security issues that transcend information systems have been fairly well researched (vid. [11] and [15]). Other areas that have been researched well have had more of a component focus. These areas include the following: VE access control (vid. [3] and [19]); authentication and integrity using certificates and password schemas; and privacy through the use of symmetric and asymmetric encryption. Research in the area of watermark technology for two-dimensional images and three-dimensional models has developed well and may hold promise for application in other areas of data integrity (vid. [1] and [2]).

One industry that has had to deal with VE security issues has been the gaming industry that needs to balance the ability to be available to anyone, yet still provide protection to the players during their gaming experience. Online VE gaming venues such as *Ultima* and *Age of Empires* have had to deal with issues such as occasional DOS attacks and the more common problem of cheaters/hackers. Some progress has been made in addressing these issues (e.g., vid. [20]).

As far as the research community goes, it has generally treated the security of VEs as an afterthought or of low priority in relation to other issues, such as performance and reliability, leaving VEs vulnerable to misuse. Until recently, systems were operated within research institutions, where developers and users had complete control of the environment, and shared a common goal of openness and information sharing. Other training venues had trusted users with complete control of hardware, software and network resources, and very limited access to VE resources by untrusted users. These research and training products were pro-

tected via physical security and simple identification and authentication (I&A) schemes. One research community that has addressed security is that of distributed computing, an area that shares many similarities with networked VEs. Particularly notable are the efforts surrounding GRID computing (e.g., vid. [8], [9], and [10]). These efforts have been primarily focused on authentication, access control, integrity, and confidentiality.

3.2 VE Security-Relevant Scenarios

Now let us turn to some examples of security-relevant scenarios to motivate the remainder of the discussion of security concerns.

Scenario no. 1: A battlefield commander is using a system that displays the area of operations and all intelligence information for that area. If the adversary in a conflict is able to gain access to the intelligence information, it could alter its own plans (e.g., move weapon assets), possibly determine and silence the source of the intelligence, or feed the system with plausible but false information (i.e., a form of information operations), with the aim of causing legitimate users to make ill-advised decisions. Alternatively, the adversary might unleash a computer virus or worm to deny use of the VE.

Scenario no. 2: Suppose an adversary can view the missions that opposing forces are training to fly in the near future, as simulated in a VE-based pre-mission trainer. The adversary could then reposition anti-aircraft platforms along the mission path to neutralize the enemy aircraft. A clever adversary might inject false objects into the trainer to confuse the pilot or cause him to attack the wrong target when he actually flies the mission. If the availability of the VE is comprised, then the pilots lose the ability to conduct visual pre-flights of the mission.

Scenario no. 3: Two companies, *A* and *B*, are competing to develop the next generation fighter aircraft for the US Navy. *A* uses a collaborative VE to design its prototype of the aircraft. If *B* can gain access to *A*'s VE without being detected, *B* can observe the successes and failures of *A*'s prototype aircraft, using this to *B*'s advantage in developing its own prototype system.

Scenario no. 4: Having detailed knowledge of a specific module used by a RTEVE, a malicious entity develops a virus that will search out that particular module in a database and modify it to contain code that will transmit every Entity State Packet Data Unit (ESPDU) received by the host application to a listening computer controlled by the entity. The attacker finds some way for the virus to be placed on the system (e.g., E-mail or an unwitting authorized user installs free software). The virus is executed and the module is modified directly within the database. A VE is then initiated which requires that specific module, and now every instantiation of the module that is downloaded by individual applications begins to transmit ESPDUs back to the listening computer. The rogue entity can observe the transactions taking place within the VE.

4. CASE STUDY: NPSNET-V

In this section we identify security concerns for RTEVEs, using NPSNET-V as a case study. This section merely provides a brief overview of the genesis and current state of NPSNET; please refer to [12] for more details. NPSNET-V is a component framework and a set of core components designed for use in the rapid construction of networked VE applications that are dynamically configurable and extensible. In this framework, all functionality—aside from a static kernel common to all applications—is provided by a set of loosely coupled modules (from network protocols to physical models) capable of being loaded and configured at runtime.

NPSNET-V makes use of tools such as Java, XML, and design patterns to achieve a number of qualities thought to be essential to a platform that is both a test bed for advanced research on VEs and a prototype for the next generation of virtual worlds. The characteristics that are sought with this design are the following:

Scalability (the ability to grow in size and scope),

Persistence (the ability for a VE to maintain a current state even when no participants are present),

Portability (the ability to run on a wide range of computing platforms),

Distribution (not rely on a single shared server or server hierarchy),

Dynamic extensibility (seamless introduction of new functionality and new classes of entities),

Interoperability (compatibility with present and future software systems), and

Composability (ability to build complex worlds from libraries of predefined reusable components).

4.1 Genesis of NPSNET-V

Bamboo [24] is similar to NPSNET-V in that Bamboo applications begin with a tiny microkernel, responsible for basic system management; the applications acquire all further functionality by loading discrete modules, or plug-ins, and connecting them to form a unified hierarchy of interdependent components. Applications can be extended by the user or triggered by an internally or externally generated event.

Like Bamboo, the Java Adaptive Dynamic Environment (JADE) is a microkernel-based component framework for RTEVEs [17]. JADE takes advantage of the Java environment to provide cross-platform support without recompilation. In JADE, networking is performed via the TreacleWell framework [18], in which network data structure and flow is controlled by the composition of discrete modules, each playing a clearly defined role in the formation, transmission, reception, and interpretation of information transmitted over the network.

Entirely textual, LambdaMOO and other MOOs are the only RTEVEs open to the public and in constant use supporting persistent virtual worlds. MOO stands for MUD, Object-Oriented; MUD stands for Multi-User Dungeon. Developed at Xerox PARC, it was used in early social virtual reality experiments [6]. Almost a decade later, it still hosts a thriving online community [14]. Participants use a simple terminal application and connect to a central server in which all processing occurs; all persistent state is stored within the server's database. Users may extend the world by adding rooms or simple objects, or they may use a specialized, prototype-based scripting language to create more advanced entities and behaviors. Users can extend pre-existing objects to create new objects and then add new behavior commands that other users can then use to interact with the new object. Another user may choose to extend this object once more, or simply clone the object to create another instance. The result, after this extension process continues for many generations, is a massive and diverse hierarchy of objects.

4.2 NPSNET-V Architecture

4.2.1 Application Structure

NPSNET-V's component framework was inspired by XML: like XML elements, NPSNET-V components are arranged in a hierarchy, with component relationships established implicitly by their relative locations within the hierarchy. An XML configuration system is an integral part of the architecture, allowing developers to combine components into functioning VE applications. The XML configuration file for the target VE application is retrieved from a lightweight directory access protocol (LDAP) persistence storage server. The file is a template the application uses to build its internal component structure; afterward the application downloads the needed modules.

NPSNET-V applications are organized using the *Model-View-Controller* pattern [13], which requires that the state model of an entity be kept separate from the representations presented to the user and the interfaces used to manipulate the model. In NPSNET-V, *models* are typically entities within the virtual world; *views* are modules that provide a graphical, textual, or auditory representation of their target entities; and *controllers* are modules that manipulate entity state in response to user input or network updates. Application layout is largely determined by functional grouping, as shown in Figure 1.

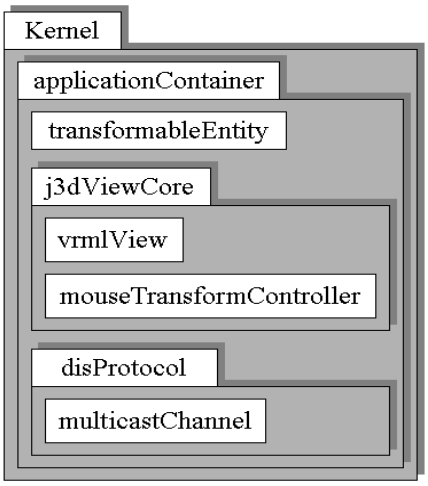

Figure 1. Example containment diagram.

In Figure 1, the white boxes indicate modules, with the shaded regions being their contents. In this example, the application contains one entity, one view core, and one protocol. The view core contains a VRML view module and a controller module that allows the user to manipulate the entity's transform using a mouse. The Distributed Interactive Simulation (DIS) protocol module, responsible for transmitting entity state over the network, contains a single multicast channel module which receives all packets generated by its containing protocol.

As the application has need to "discover" new modules or entities, it will communicate with the LDAP server and retrieve a URL that points to the desired module. It then contacts the appropriate HTTP server and downloads the module, assimilating it into its internal structure. This provides the runtime extensibility of NPSNET-V.

4.2.2 Database Structure

The database architecture is divided into two groups: persistent storage and custom servers. Messages exchanged between peers, particularly those associated with the Dynamic Behavior Protocol (DBH), may cause the receiving protocol module to extend the application hierarchy by loading and activating new modules. For instance, receiving a message addressed to an entity that does not yet exist on the host system will cause a representation of that entity to be loaded. These representations must be easily accessible by all of the participants of a virtual world.

NPSNET-V uses LDAP and HTTP servers to provide this functionality. The LDAP service, which may be distributed among multiple LDAP servers, provides a unified namespace for virtual-world components. Each entry in the directory contains an XML configuration that represents a serialized part of the VE application; the URLs for the necessary application modules are included.

In general, these modules are stored within Java archives (jars) along with extended meta data concerning their contents and made accessible via the HTTP servers. The component framework downloads and caches these jars as necessary to provide the resources required by hosted applications.

In some cases, HTTP access is not appropriate, and specialized protocols must be used. For instance, an example application uses a custom interface to download elevation data from a specialized terrain server. In addition, streaming video and audio require their own custom interface.

4.2.3 Network Structure

Figure 2 depicts an example of the network connections that may be formed during a typical session. The dotted lines indicate unreliable multicast channels used to transfer entity state updates and interactions between peers. Solid lines represent the transient

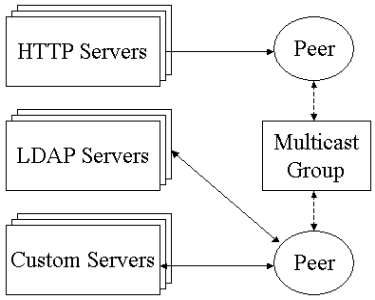

Figure 2. Overview of network

reliable connections used to download resources from the World Wide Web, to store and retrieve configuration data from the LDAP service, and to access custom servers.

Three types of communication flow through this network, one being administrative communication, such as retrieving configuration files and URLs, generally requiring reliable TCP or multicast connections. A second type of flow is that of peer-to-peer, packet-based state updates and interactions transmitted over IP multicast, the principle mode of communication between individual applications. In the simplest case, one multicast channel may be used for all network traffic within a world, an impractical approach for worlds with many participants. *Area-of-interest (AOI) management* is used to distribute traffic among many multicast channels according to specified criteria. A third type of flow,

module/resource communication, requires reliable TCP/UDP connections. These flows include passing of module code, terrain data, and streaming video and audio from HTTP and custom servers.

4.3 Taxonomy of RTEVE Security Issues

Here we develop the beginnings of a taxonomy of security issues by grouping the concerns into the following categories: integrity, confidentiality, availability, non-repudiation, and authentication; see Table 1. (N.B.: This article does not cover the attacks that may be possible on the workstation itself, or attacks on the TCP/IP stack. These subjects are outside the scope of this paper, but have been extensively studied by others.)

4.3.1 Integrity

There are a number of possible attacks on the integrity of the RTEVE system, many of them particularly dangerous because information that has traditionally been kept locally on systems presumed to be secure may be loaded across the network, where it is vulnerable to interception and modification. These can be divided into attacks on configuration files, components, data communications, the database, and temporal integrity.

In NPSNET-V, it will be necessary to protect XML configuration files from unauthorized modification by users who want to direct an application to add in a module containing malicious code.

Since the RTEVE application itself could be modified by loading new components at runtime, a Trojan horse could be substituted for a legitimate component in transmission, resulting in compromised data or modified behavior of the RTEVE. The components added to the virtual world (as opposed to the RTEVE administrative framework) are likewise potential targets of worms or Trojan horses.

Once the components have been loaded, they can communicate with each other. This presents another avenue for an attacker. The communications can be subverted, perhaps to supply false but plausible entity state data that confuse or deceive others in the virtual world. Data might also be modified in targeted ways to cause the VE to reveal information in an unintended way, or to cause a system crash, or to execute a buffer overflow attack. In general, the class of communications attacks is very similar to those that current, non-RTEVEs face.

In addition, components could be subverted while in the database. An attacker could replace legitimate components with malicious ones, and then wait for an unsuspecting user to load them. In older, static VEs this information was often distributed in a secure manner (at least conceptually) before the application started and kept on a presumed secure local machine. A *Quake* user, for example, has a database of levels, textures, entities, code, etc.; he would receive and load this data from a CD onto the hard disk of his workstation. The application would be as secure as it was originally distributed. Now, if a malicious user was able to break into a server that contained an update to the application, and replace the update with one that contained say a virus or worm; then as users downloaded the update, they in reality would be downloading a Trojan horse, and their application would no longer be secure, in fact it would now be subverted with whatever payload the Trojan horse contained. With a RTEVE, this scenario could occur at any time during runtime. A user could begin with a pristine, secure copy of the application, but then if a malicious module were to be assimilated, the application would be compromised. Depending on the nature of the malicious code, the user's application/workstation would be attacked, or used in an attack such as a DOS attack.

Moreover, the Network Time Protocol (NTP) synchronization information provided by the NTP subsystem could be modified to disrupt the synchronization of the participants in a RTEVE, reducing or preventing the usability of the VE for interactions. The distortion of temporal data can be used to reorder events, perhaps allowing an "effect" to occur before a "cause."

4.3.2 Confidentiality

Although NPSNET-V is a research prototype, it is intended to be a blueprint for RTEVEs used by organizations, such as the U.S. Army, which have policies that govern the protection of sensitive information. Within this context, it will be necessary to maintain secrecy of information on both the network and individual workstations. Likewise, it will need an access control mechanism to ensure that only authorized participants can access data and computing resources. The RTEVE also needs to protect application code on a workstation from being surreptitiously acquired; such acquired code could be used in developing future attacks or modified malicious copies that are then substituted for the originals.

Information can be gleaned by the analysis of traffic flowing across the network. Depending on the purpose of the application and current state of affairs, an increase in data transmission can signal preparations for a military action or reveal relationships between units. Even if the data is not readable, an attacker can determine which participants are most active or most informed and use that information to target the resources being used by those participants.

4.3.3 Availability

The availability of a RTEVE could be of great importance to an organization that is using the system to complete a time-critical task such as a rehearsal for an air strike. RTEVEs are distributed by their very nature and are thus potential targets of malicious users who initiate distributed DOS attacks: this includes any type of network or computer attack designed to reduce or eliminate the usability of a RTEVE, such as packet saturation of the communication paths or introduction of code designed to shut down one or more applications. Focused DOS attacks could be directed toward physical components of the RTEVE, such as data links or workstations.

4.3.4 Non-Repudiation

Returning to an example we presented is Section 3, a commander that monitors a battle space through a RTEVE must be able to positively identify the source of data which he uses to make a decision. If he gives the order to launch missiles on what the system shows is a hostile aircraft, but in reality is a passenger jetliner, the source of the information must be traceable and must not be able to repudiate its introduction of the misinformation.

4.3.5 Authentication

A malicious entity could use methods such as IP hijacking or spoofing to enter into the RTEVE and observe and interact with participants. There is also the possibility that a malicious entity possesses a copy of the application and is able to be accepted as an authorized participant. NPSNET-V will need to be able to authenticate users across a distributed computing environment, necessitating for example, the incorporation of a public-key infrastructure (PKI) into the architecture.

4.4 Information Assurance

The foregoing taxonomy, although incomplete, is a starting point for both building a more inclusive taxonomy and making sure that major areas of concern regarding information assurance are considered when developing or maintaining RTEVEs. In some cases there are two or more ways of addressing a security concern: the decision of which alternative to use can be driven by the results of performing a systematic assessment of the security policy and requirements of the RTEVE. As part of the information assurance activities, a risk assessment of the RTEVE would need to be conducted: vulnerabilities are not a concern unless there is a threat associated with the vulnerabilities, and the likelihood and magnitude of the loss is likely to exceed the threshold set by a user or owner of the RTEVE. There needs to be some level of agreement among participants in a RTEVE about the acceptable threshold.

When formulating security policy for a RTEVE, the developer must balance the security measures needed to enforce such policy with other requirements, such as QOS. For example, addressing non-repudiation of data-update packets by using certificates could make it difficult to satisfy a guarantee that a delay in transmission will not exceed an agreed on threshold value, preventing the effective use of the VE in a real-time context.

Information assurance activities would need to be performed throughout the life cycle of the RTEVE: the system is non-stationary in that components are continually being added and removed from the system, and it could be used in a wide spectrum of operational contexts, from those that have very minimal security requirements to those in which the compromise of the RTEVE could contribute to the demise of an organization or even have an adverse affect on national security.

4.5 Status of NPSNET-V Security

NPSNET-V is a Java-based application with no security beyond the default provided by the Java Virtual Machine (JVM). We are developing security policy and requirements for NPSNET-V using, as a guide, both the architectural framework described in Section 4.2 and the taxonomy presented in Section 4.3. The goal is to provide adequate security with minimal reduction in the flexibility of the architecture.

4.5.1 General

An easy assumption upon which to construct networked applications is that any security concern can generally be resolved via existing computer, network and database security mechanisms. This assumption requires that the VE be hosted on a secure network. For an application such as NPSNET-V, whose goal is to be as flexible as possible, this assumption cannot be accepted, nor the assumption that any system it resides on will have computer and database security measures already in place; not to mention that the application must be able to bridge both secure and non-secure networks. Therefore, the desired security level of the application must be ensured by the application itself. If the application were on an untrusted system, then it would at least be protected to the extent that is inherent in the application itself; if, on the other hand, the application is on a secure system, then the application is that much more protected.

The GRID security infrastructure mentioned in Section 3.1 has merit for possible use with a RTEVE. The application would need to be designed to work on a GRID infrastructure, or the GRID infrastructure would need to be modified to work with the application. The GRID infrastructure already takes into account authentication, access control, confidentiality, and integrity. We suggest, however, that the integrity model used (i.e., certificates) is insufficient if the data is maliciously modified prior to communication (e.g., by a Trojan horse or direct modification). In the case of transferring executable code to possibly thousands of participants, it is essential that a malicious modification be identified, no matter when it occurred.

4.5.2 Current Ideas

Current ideas revolve around the creation of a Distributed Security Manager (DSM) that is based on complementing layers, similar to that discussed in [7]. The DSM would mange security strictly within the application. Identified layers for NPSNET-V are: encryption, I&A, and intrusion/misuse detection.

Encryption. Use of PKI and conventional keys through the Java Secure Socket Layer (SSL) to maintain encryption of data within the network. The DSM should manage the dynamic redistribution of keys in response to breaches of security.

I&A. Used to ensure the legitimacy of connecting applications. Most likely the implementation will require the use of PKI certificates to verify identity.

Intrusion/Misuse Detection. There must be some way to identify compromised modules and the presence of malicious users/observers, as well as simple and complex/distributed network attacks. Current design ideas are based on research into both signature- and anomaly-based network intrusion detection systems (IDS) such as [7], [22] and, [23]. Our thoughts include the use of hashing signatures, or perhaps even some form of component watermarking, as a way to identify modified modules, and audit logs to identify malicious activity.

We envision the DSM to have a distributed nature with some form of security module in each application that communicates with either several "senior" security modules that provide high-level oversight, or each other for cooperative oversight. The main ideas here are to both avoid a single point of failure and allow for system robustness in case some security modules are compromised. The DSM should also be designed to avoid its use as an unwitting vehicle for DOS attacks. This could occur if an attacker continuously performs a simple action that causes the DSM to keep redistributing cryptography keys, thus causing the VE to halt as keys are continually changing.

The realization of the DSM will likely require modifications to be made to the current NPSNET-V architecture and the initial taxonomy of security concerns presented in the article.

5. CONCLUSION

The complexity and size of RTEVEs opens them to a wide spectrum of security vulnerabilities, in addition to those found in any interactive distributed application. As RTEVEs make their way outside the laboratory and into mainstream usage, additional security-related vulnerabilities will arise. At present, the authors of this paper are exploring the use of combination of existing and novel information assurance techniques to eliminate or mitigate the effects of such vulnerabilities.

6. ACKNOWLEDGEMENTS

We thank Don Brutzman for his comments on earlier drafts of this article. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

7. REFERENCES

[1] Benedens, O. Geometry-based watermarking of 3D models. *IEEE Computer Graphics and Applications*, 19, 1 (Jan./Feb. 1999), 46-55.

[2] Berghel, H. Watermarking Cyberspace. *Comm. ACM*, 40, 11 (Nov. 1997), 19-24.

[3] Bullock, A. and Benford, S. An access control framework for multi-user collaborative environments. In *Proc. Int. SIGGROUP Conf. Supporting Group Work*, ACM (Phoenix, Ariz., Nov. 1999), 140-149.

[4] Capps, M., McGregor, D., Brutzman, D., and Zyda, M. NPSNET-V: A New beginning for dynamically extensible virtual environments. *IEEE Computer Graphics and Applications* 20, 5 (Oct. 2000), 12-15.

[5] Capps, M. and Stotts, D. Research issues in developing networked virtual realities: Working group report on distributed system aspects of sharing a virtual reality. In *Proc. Sixth Workshop Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE (Cambridge, Mass., June 1997), 205-211.

[6] Curtis, P. and Nichols, D. MUDs grow up: Social virtual reality in the real world. In *Digest of Papers for COMPCON*, IEEE (San Francisco, Calif., Feb. 1994), 193-200.

[7] Forrest, S., Hofmeyr, S., and Somayaji, A. Computer immunology. *Comm. ACM* 40, 3 (Oct. 97), 88-96.

[8] Foster, I., Karonis, T., and Kesselman, C. Managing security in high-performance distributed computations. *Cluster Computing* 1, 1 (1998), 95-107.

[9] Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S. A security architecture for computational grids. In *Proc. Fifth Conf. Computer and Communications Security*. ACM (San Francisco, Calif., Nov. 1998), 83-92.

[10] Foster, I., Kesselman, C., and Tuecke, S. The anatomy of the grid: enabling scalable virtual organizations. *Int. J. Supercomputer Applications* 15, 3 (Fall 2001), 200-222.

[11] Jayaram, N. D. and Morse, P. L. R. Network security: A taxonomic view. In *European Conf. Sec. and Detection*, IEEE (London, Apr. 1997), 124-127.

[12] Kapolka, A., McGregor, D., and Capps, M. A unified component framework for dynamically extensible virtual environments. In *Proc. Fourth Int. Conf. Collaborative Virtual Environments*, ACM (Bonn, Germany, Sept. 2002).

[13] Krasner,G., and Pope, S. A cookbook for using the model-view-controller user interface paradigm in smalltalk-80. *J. Object-Oriented Prog.* 1, 3 (Aug./Sep. 1988), 26-41.

[14] LambdaMOO. <telnet://lambda.moo.mud.org:8888>

[15] Landwehr, C. E., Bull A., McDermott, J., and Choi, W. A taxonomy of computer program security flaws. *ACM Comput. Surveys* 26, 3 (Sept. 1994), 211-254.

[16] Macedonia, M. R. and Zyda, M. J. A taxonomy for networked virtual environments. *IEEE Multimedia* 4, 2 (Jan./Mar. 1997), 48-56.

[17] Oliveira, M., Crowcroft, J., and Slater, M. Component framework infrastructure for virtual environments. In *Proc. Third Int. Conf. Collaborative Virtual Environments*, ACM (San Francisco, Calif., Sept. 2000), 139-146.

[18] Oliveira, M., Crowcroft, J., Brutzman, D., and Slater, M. Components for distributed virtual environments. In *Proc. Symposium Virtual Reality Softw. and Technol.*, ACM (London, Dec. 1999), 176-177.

[19] Pettifer, S. and Marsh, J. Collaborative access model for shared virtual environments. In *Proc. Tenth Int. Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE (Cambridge, Mass, June 2001), 257-262.

[20] Pritchard, M. How to hurt the hackers: the inside scoop on internet cheating and how you can combat it. *Game Developer* 7, 6 (June 2000), 28-40.

[21] Singhal, S. and Zyda, M. *Networked Virtual Environments: Design and Implementation*. ACM Press-SIGGRAPH Series, New York, 1999.

[22] Stillerman, M., Marceau, C., and Stillman, M. Intrusion detection for distributed applications. *Comm. ACM* 42, 7 (July 1999), 62-69.

[23] Vigna, G. and Kemmerer, R. NetStat: A network-based intrusion detection approach. In *Proc. Fourteenth Annual Computer Sec. Application Conf.*, ACM (Scottsdale, Ariz., Dec. 1998), 25-34.

[24] Watsen, K. and Zyda, M. Bamboo: A portable system for dynamically extensible, real-time, networked, virtual environments. In *Proc. Virtual Reality Annual Int. Symposium*, IEEE (Atlanta, Ga., Mar. 1998), 252-259.